Exhibit 10 to Complaint
Intellectual Ventures I LLC and Intellectual Ventures II LLC

Example American Count 3 Systems and Services
U.S. Patent No. 7,949,785 ("the '785 Patent")

The Accused Systems and Services include without limitation American systems and services that utilize Kubernetes; all past, current, and future systems and services that operate in the same or substantially similar manner as the specifically identified systems and services; and all past, current, and future American systems and services that have the same or substantially similar features as the specifically identified systems and services ("Example American Count 3 Systems and Services" or "American Systems and Services").[1]

On information and belief, the American Systems and Services use Kubernetes in its private cloud(s). For example, American posts, or has posted, job opportunities that require familiarity with Kubernetes containerization concepts.

- Example of job listing for an Engineer/Sr Engineer, IT Situational Awareness at American Airlines which requires proficiency in Kubernetes. https://jobs.aa.com/job/EngineerSr-Engineer%2C-IT-Situational-Awareness/75837-en_US (Last accessed on 10/31/2024).
- Example of job listing for an Associate Developer, IT Applications at American Airlines which requires proficiency in Kubernetes. https://jobs.aa.com/job/Associate-Developer%2C-IT-Applications/75816-en_US (Last accessed on 10/31/2024).
- Example of Senior Java Full Stack Developer position at American Airlines which mentions use of Kubernetes. https://www.linkedin.com/in/rohitha-m6363/ (Last accessed on 9/19/24)
- Example of Sr. Kubernetes Engineer/Architect position at American Airlines which mentions heavy use of Kubernetes. https://www.linkedin.com/in/sudheer-patchari/ (Last accessed on 9/19/24)
- Example of Kubernetes Engineer position at American Airlines which mentions heavy use of Kubernetes. https://www.linkedin.com/in/sridhar-pulluri-199b56250/ (Last accessed on 9/19/24)
- Example of Sr. Cloud Infra DevSecOps Engineer/Architect position at American Airlines which mentions use of Kubernetes. https://www.linkedin.com/in/rupa-m-b90836309/ (Last accessed on 9/19/24)
- Example of DevOps Engineer position at American Airlines which mentions use of Kubernetes. https://www.linkedin.com/in/manidhar-a-555726169/.
- Example of Java Developer position at American Airlines which mentions use of Kubernetes. https://www.linkedin.com/in/suganya-koodalingam-8a0590102/.

[1] For the avoidance of doubt, Plaintiffs do not accuse public clouds of American if those services are provided by a cloud provider with a license to Plaintiffs' patents that covers American's activities. IV will provide relevant license agreements for cloud providers in discovery. To the extent any of these licenses are relevant to American's activities, Plaintiffs will meet and confer with American about the impact of such license(s).

- Example of Senior Java Developer position at American Airlines which mentions use of Kubernetes. https://www.linkedin.com/in/ganesh-kenda/

As another example, American has announced cloud migration of legacy technology and efforts to modernize its mainframes and servers. Source: https://dxc.com/sg/en/insights/customer-stories/american-airlines-cloud-data-automation.

On information and belief, additional information confirms American uses Kubernetes technology.

Q2: You've been leading digital transformation and modernization initiatives for a while, in very complex, demanding organizations. Looking back at your career, what are some of the challenges (organizational, business, and technical) that you encountered with modernization initiatives?

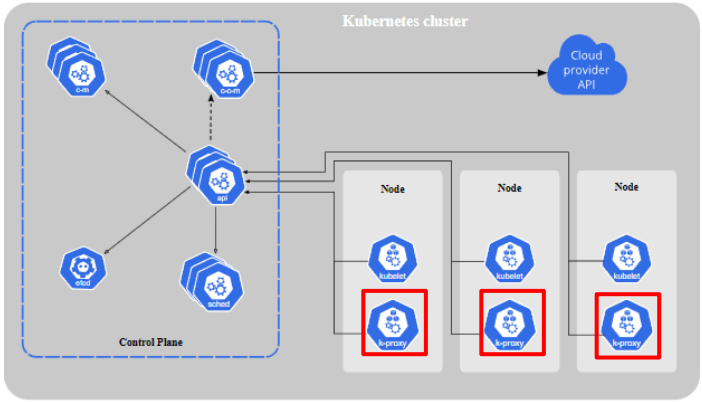
Jason: In late 2018, we started our digital transformation by creating a tenancy in Oracle Cloud Infrastructure (OCI) and iterating through multiple proofs of technology, first proving out that our on-premises applications could not only run functionally, but also run workloads as fast or even faster in OCI. Shortly after that, a core group of infrastructure team members (led by our brilliant technical delivery manager, Vijay Krishnaswamy) began learning the fundamentals of Kubernetes and related open source software. Despite their team having success migrating their applications into OCI in 2020 and then redeploying their applications into Kubernetes a year later, we faced a larger challenge with the team supporting Ventana, the core application that supports the AAdvantage program. The stakes were huge: Ventana is one of American Airlines' most important applications. Additionally, the team supporting it had not been directly involved in the journey forged by the other group.

When we merged both teams into one organization, encouraged team members to learn from and teach each other, and demonstrated sustained leadership support, we emerged with a crossfunctional team who achieved multiple certifications in both OCI and Kubernetes. They were then equipped with the skills, confidence, and support to move Ventana into not just OCI, but also Oracle Container Engine for Kubernetes (OKE) in one go.

Source: https://blogs.oracle.com/cloud-infrastructure/post/five-questions-jason-maczura-american-airlines. [2]

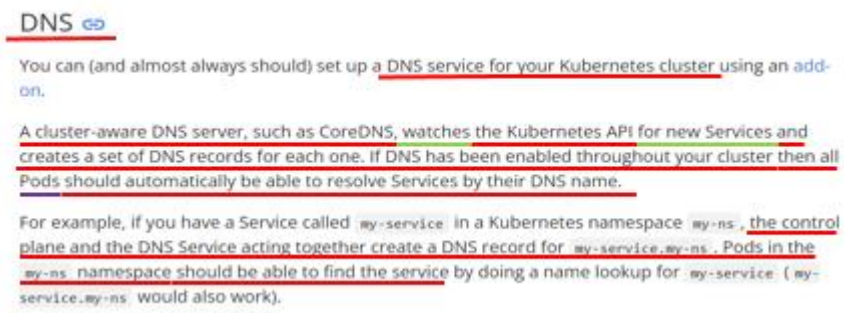
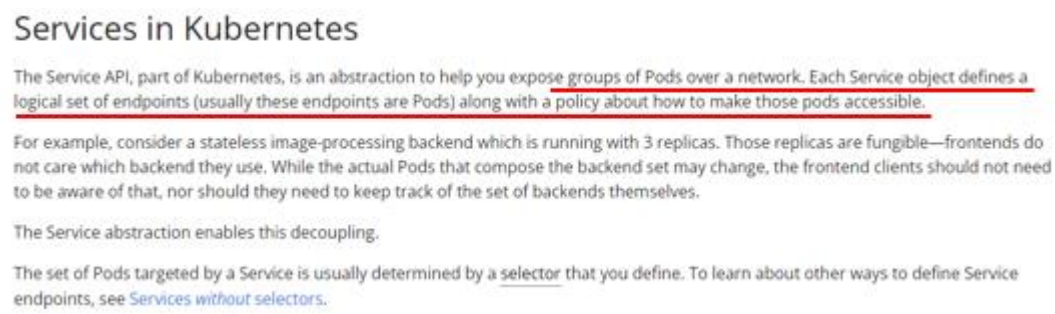
[2] All sources cited in this document were publicly accessible as of the filing date of the Complaint.

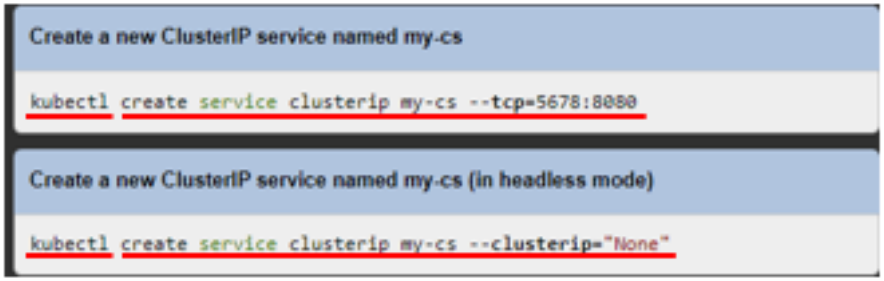
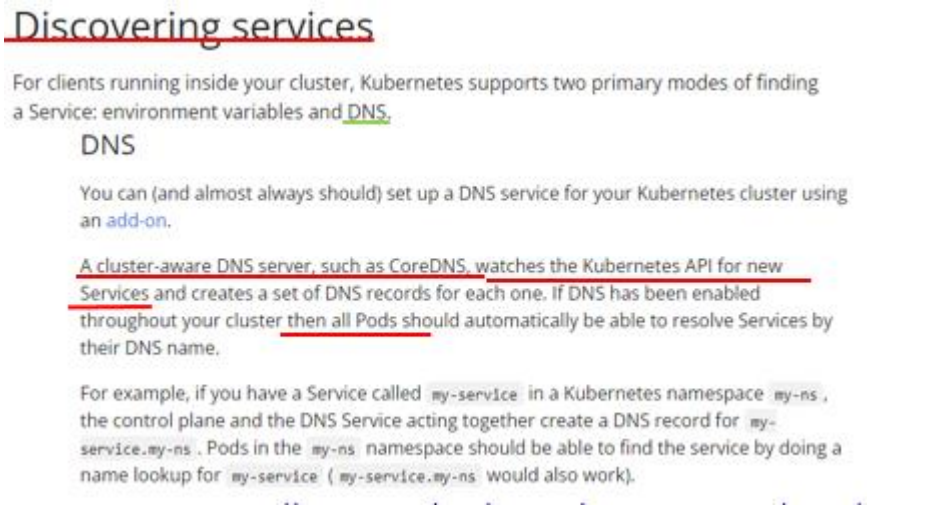
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
[30.pre] A virtual network manager, comprising:	To the extent this preamble is limiting, on information and belief, the American Count 3 Systems and Services include a virtual network manager. The virtual network manager is the Kubernetes functionality related to DNS for Services and Pods. **DNS for Services and Pods** Kubernetes creates DNS records for Services and Pods. You can contact Services with consistent DNS names instead of IP addresses. Kubernetes publishes information about Pods and Services which is used to program DNS. Kubelet configures Pods' DNS so that running containers can lookup Services by name rather than IP. Services defined in the cluster are assigned DNS names. By default, a client Pod's DNS search list includes the Pod's own namespace and the cluster's default domain. Source: https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/#pod-s-dns-config. A Service in Kubernetes is an abstraction which defines a logical set of Pods and a policy by which to access them. Services enable a loose coupling between dependent Pods. A Service is defined using YAML or JSON, like all Kubernetes object manifests. The set of Pods targeted by a Service is usually determined by a *label selector* (see below for why you might want a Service without including a `selector` in the spec). Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/. Kubernetes assigns this Service an IP address (the *cluster IP*), that is used by the virtual IP address mechanism. For more details on that mechanism, read Virtual IPs and Service Proxies. Source: https://kubernetes.io/docs/concepts/services-networking/service/.

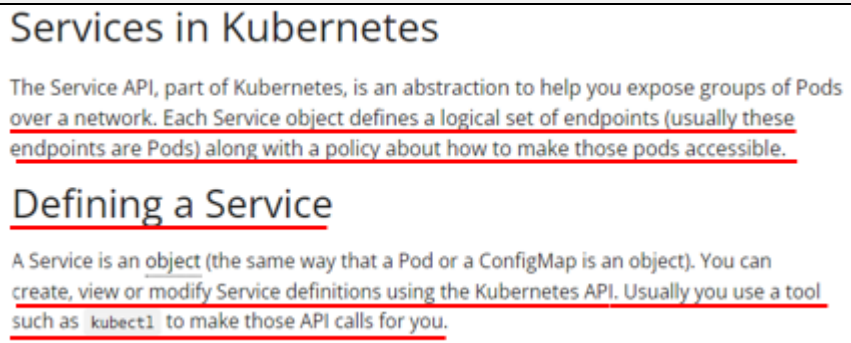
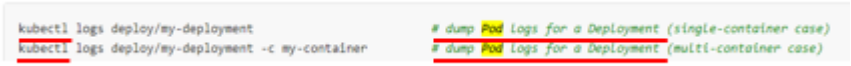
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	Virtual IPs and Service Proxies Every node in a Kubernetes cluster runs a kube-proxy (unless you have deployed your own alternative component in place of `kube-proxy`). The `kube-proxy` component is responsible for implementing a *virtual IP* mechanism for Services of `type` other than `ExternalName`. Source: https://kubernetes.io/docs/reference/networking/virtual-ips/.
[30.a] a network interface configured for data communication via a virtual network that is defined by a domain name having an associated public network address;	On information and belief, the American Count 3 Systems and Services include a network interface configured for data communication via a virtual network that is defined by a domain name having an associated public network address. On information and belief, Kubernetes DNS for Services and pods is implemented in part through a network interface called kube-proxy, which maintains network rules that allow communication to pods from network sessions. Kube-proxy communications can relate to network sessions inside or outside of the cluster in which the services are defined. The services defined in the cluster are assigned DNS names having an associated public network address. **Kubernetes Components** 

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	kube-proxy kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster. Source: https://kubernetes.io/docs/concepts/overview/components/#:~:text=kube%2Dproxy%20is%20a%20network,or%20outside%20of%20your%20cluster. DNS A cluster-aware DNS server, such as CoreDNS, watches the Kubernetes API for new Services and creates a set of DNS records for each one. If DNS has been enabled throughout your cluster then all Pods should automatically be able to resolve Services by their DNS name. External IPs If there are external IPs that route to one or more cluster nodes, Kubernetes Services can be exposed on those `externalIPs`. When network traffic arrives into the cluster, with the external IP (as destination IP) and the port matching that Service, rules and routes that Kubernetes has configured ensure that the traffic is routed to one of the endpoints for that Service. Source: https://kubernetes.io/docs/concepts/services-networking/service/.

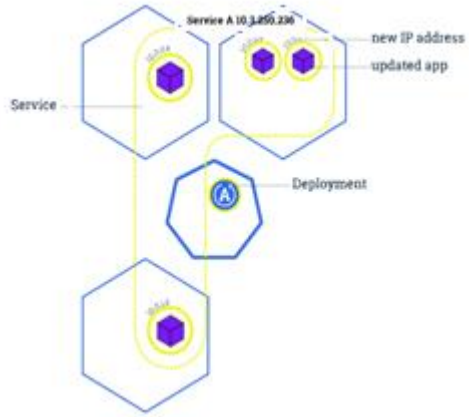
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	DNS for Services and Pods Kubernetes creates DNS records for Services and Pods. You can contact Services with consistent DNS names instead of IP addresses. Services defined in the cluster are assigned DNS names. By default, a client Pod's DNS search list includes the Pod's own namespace and the cluster's default domain. Source: https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/#pod-s-dns-config.
[30.b] a memory and a processor to implement a register module configured to register devices in a virtual network, the register module further configured to:	On information and belief, the American Count 3 Systems and Services include a memory and a processor to implement a register module configured to register devices in a virtual network. On information and belief, a service runs on a set of pods. The Kubernetes DNS Service module watches the Kubernetes API for incoming new services and creates a set of DNS records for each of the pods associated with the service. Kubernetes runs your workload by placing containers into Pods to run on *Nodes*. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods. Source: https://kubernetes.io/docs/concepts/architecture/nodes. Initializing your control-plane node The control-plane node is the machine where the control plane components run, including etcd (the cluster database) and the API Server (which the kubectl command line tool communicates with). Source: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/.

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 DNS You can (and almost always should) set up a DNS service for your Kubernetes cluster using an add-on. A cluster-aware DNS server, such as CoreDNS, watches the Kubernetes API for new Services and creates a set of DNS records for each one. If DNS has been enabled throughout your cluster then all Pods should automatically be able to resolve Services by their DNS name. For example, if you have a Service called `my-service` in a Kubernetes namespace `my-ns`, the control plane and the DNS Service acting together create a DNS record for `my-service.my-ns`. Pods in the `my-ns` namespace should be able to find the service by doing a name lookup for `my-service` (`my-service.my-ns` would also work). Source: https://kubernetes.io/docs/concepts/services-networking/service.  Services in Kubernetes The Service API, part of Kubernetes, is an abstraction to help you expose groups of Pods over a network. Each Service object defines a logical set of endpoints (usually these endpoints are Pods) along with a policy about how to make those pods accessible. For example, consider a stateless image-processing backend which is running with 3 replicas. Those replicas are fungible—frontends do not care which backend they use. While the actual Pods that compose the backend set may change, the frontend clients should not need to be aware of that, nor should they need to keep track of the set of backends themselves. The Service abstraction enables this decoupling. The set of Pods targeted by a Service is usually determined by a selector that you define. To learn about other ways to define Service endpoints, see Services *without* selectors.
[30.b.i] receive a registration request from an agent associated with a device;	On information and belief, the American Count 3 Systems and Services include a register module configured to receive a registration request from an agent associated with a device. On information and belief, a service runs on a set of pods. The Kubernetes DNS Service module watches the Kubernetes API for incoming new services being made from the kubelet.

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 Source: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands.  Source: https://kubernetes.io/docs/concepts/services-networking/service.

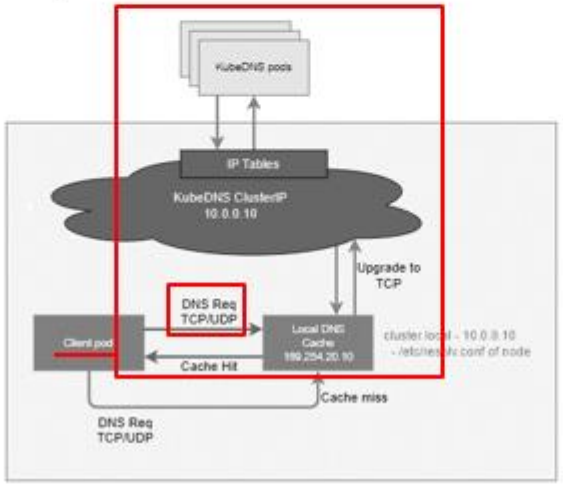
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/services-networking/service.  Source: https://kubernetes.io/docs/reference/kubectl/cheatsheet/. See also https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/.

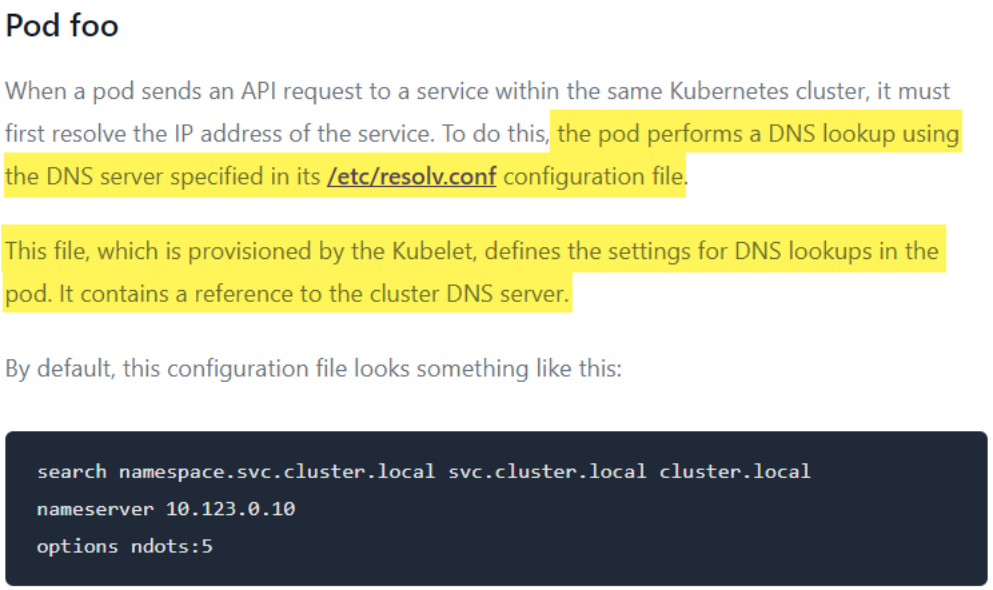
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
[30.b.ii] distribute a virtual network address to the device when the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network address; and	On information and belief, the American Count 3 Systems and Services include a register module configured to distribute a virtual network address to the device when the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network address. On information and belief, a service runs on a set of pods. A new service is assigned to a cluster IP, and as part of configuration, the Kubernetes DNS Service module distributes the network address to at least one pod on which the service runs. **Using a Service to Expose Your App** Although each Pod has a unique IP address, those IPs are not exposed outside the cluster without a Service. Services allow your applications to receive traffic. Services can be exposed in different ways by specifying a `type` in the `spec` of the Service: • *ClusterIP* (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster. Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/ (last accessed on November 10, 2023) **Service ClusterIP allocation** In Kubernetes, Services are an abstract way to expose an application running on a set of Pods. Services can have a cluster-scoped virtual IP address (using a Service of `type: ClusterIP`). Clients can connect using that virtual IP address, and Kubernetes then load-balances traffic to that Service across the different backing Pods. Source: https://kubernetes.io/docs/concepts/services-networking/cluster-ip-allocation/ (last accessed on November 10, 2023) **VIP** a virtual IP address, such as the one assigned to every Service in Kubernetes

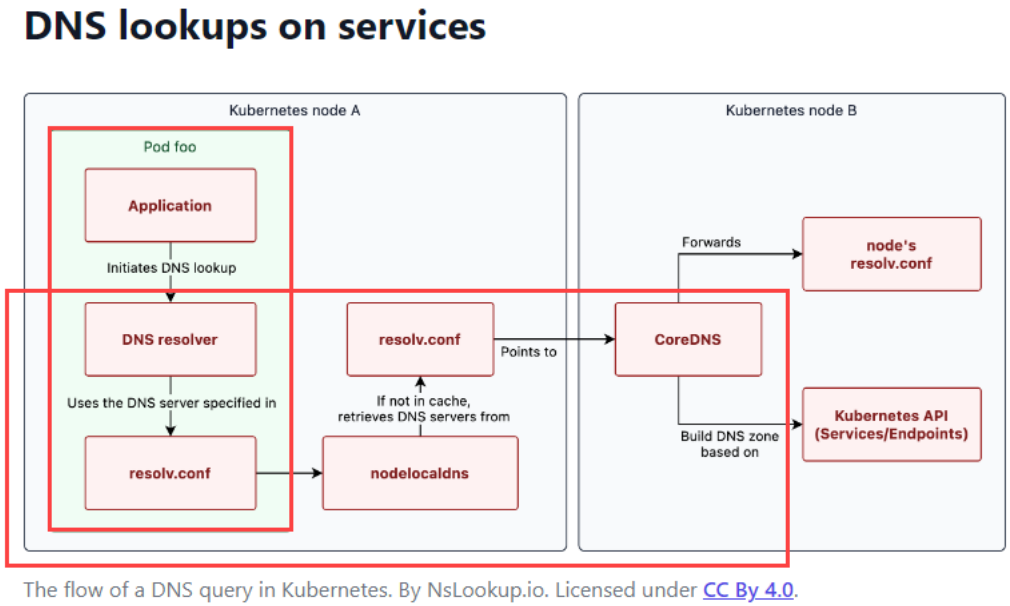
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	Source: https://kubernetes.io/docs/tutorials/services/source-ip/ (last accessed on November 10, 2023) DNS You can (and almost always should) set up a DNS service for your Kubernetes cluster using an add-on. A cluster-aware DNS server, such as CoreDNS, watches the Kubernetes API for new Services and creates a set of DNS records for each one. If DNS has been enabled throughout your cluster then all Pods should automatically be able to resolve Services by their DNS name. For example, if you have a Service called `my-service` in a Kubernetes namespace `my-ns`, the control plane and the DNS Service acting together create a DNS record for `my-service.my-ns`. Pods in the `my-ns` namespace should be able to find the service by doing a name lookup for `my-service` (`my-service.my-ns` would also work). Source: https://kubernetes.io/docs/concepts/services-networking/service (last accessed on November 10, 2023)  Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/

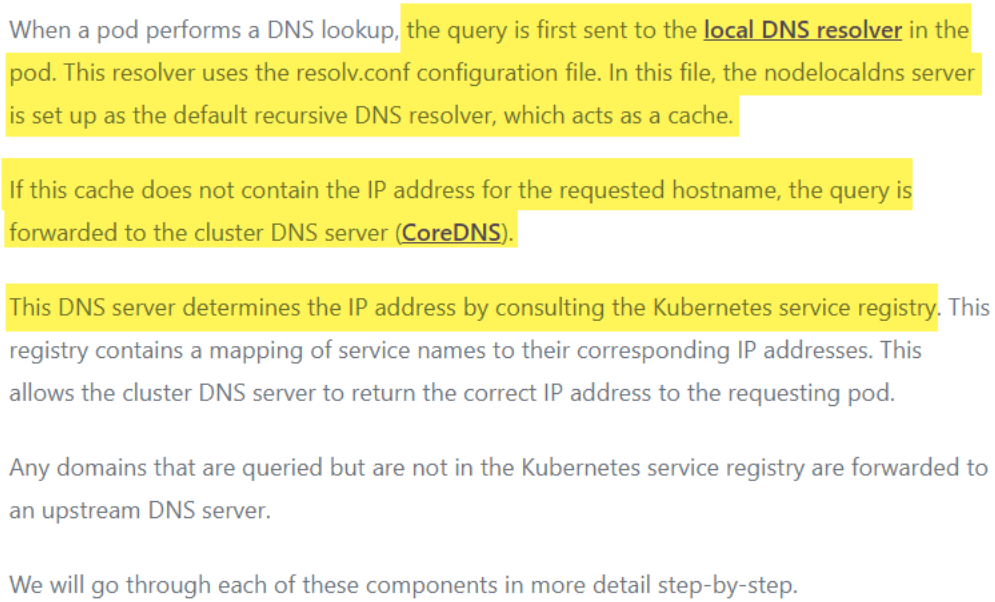
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
[30.c] a DNS server for the virtual network, the DNS server configured to receive a DNS request from a first device in the virtual network and return a network address associated with a network route director, a private network address associated with a second device in the virtual network, and a virtual network address associated with the second device.	On information and belief, the American Count 3 Systems and Services include a DNS server for the virtual network, the DNS server configured to receive a DNS request from a first device in the virtual network and return a network address associated with a network route director, a private network address associated with a second device in the virtual network, and a virtual network address associated with the second device. On information and belief, a client / frontend pod sends a DNS query using Kubernetes DNS query functionality, such as CoreDNS or kube-dns, which includes resolver routines associated with the requesting pod. The DNS server (cluster DNS server, e.g. CoreDNS or kube-dns) is configured to return an IP address of cluster DNS server, which is referenced/returned by the process of the kubelet accessing resolv.conf. resolv.conf(5) File Formats Manual resolv.conf(5) NAME resolv.conf - resolver configuration file SYNOPSIS /etc/resolv.conf DESCRIPTION The resolver is a set of routines in the C library that provide access to the Internet Domain Name System (DNS). The resolver configuration file contains information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information. The configuration file is considered a trusted source of DNS information; see the trust-ad option below for details. Source: https://man7.org/linux/man-pages/man5/resolv.conf.5.html (last accessed on November 10, 2023)

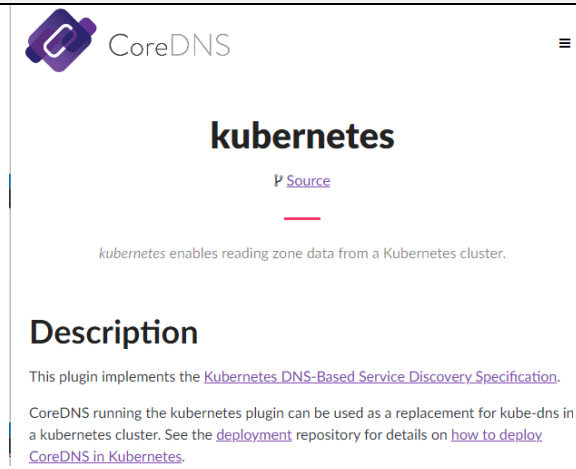
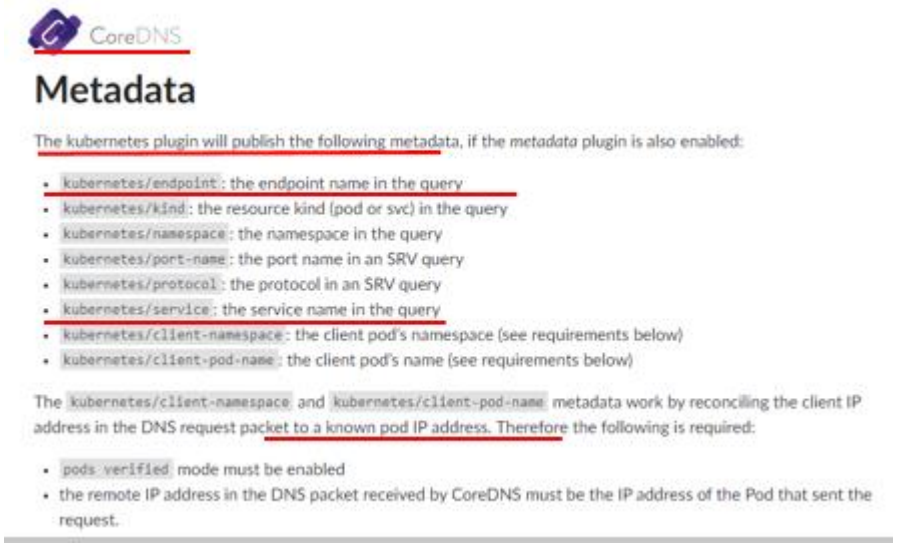
U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 Source: https://kubernetes.io/docs/tasks/administer-cluster/coredns/ (last accessed on November 10, 2023)

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	Introduction NodeLocal DNSCache improves Cluster DNS performance by running a DNS caching agent on cluster nodes as a DaemonSet. In today's architecture, Pods in 'ClusterFirst' DNS mode reach out to a kube-dns `serviceIP` for DNS queries. This is translated to a kube-dns/CoreDNS endpoint via iptables rules added by kube-proxy. With this new architecture, Pods will reach out to the DNS caching agent running on the same node, thereby avoiding iptables DNAT rules and connection tracking. The local caching agent will query kube-dns service for cache misses of cluster hostnames (" `cluster.local` " suffix by default).  Source: https://kubernetes.io/docs/tasks/administer-cluster/nodelocaldns/ (last accessed on November 10, 2023)

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 Source: https://www.nslookup.io/learning/the-life-of-a-dns-query-in-kubernetes/.

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 The flow of a DNS query in Kubernetes. By NsLookup.io. Licensed under CC By 4.0. Source: https://www.nslookup.io/learning/the-life-of-a-dns-query-in-kubernetes/.

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	 Source: https://www.nslookup.io/learning/the-life-of-a-dns-query-in-kubernetes/. On information and belief, American's systems include "a private network address associated with a second device in the virtual network, and a virtual network address associated with the second device." On information and belief, the private network address associated with a second device is a unique private IP address assigned to a backend node. The virtual network address associated with the second device is the Cluster IP or Virtual IP assigned to a pod (or pods) and/or container (or containers). .

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	  Source: https://coredns.io/plugins/kubernetes//.

U.S. Patent No. 7,949,785 (Claim 30)	
Claim(s)	**Example American Count 3 Systems and Services**
	Connecting Applications with Services Kubernetes assumes that pods can communicate with other pods, regardless of which host they land on. Kubernetes gives every pod its own cluster-private IP address, so you do not need to explicitly create links between pods or map container ports to host ports. This means that containers within a Pod can all reach each other's ports on localhost, and all pods in a cluster can see each other without NAT. The rest of this document elaborates on how you can run reliable services on such a networking model. Source: https://kubernetes.io/docs/tutorials/services/connect-applications-service/. **Using a Service to Expose Your App** Although each Pod has a unique IP address, those IPs are not exposed outside the cluster without a Service. Services allow your applications to receive traffic. Services can be exposed in different ways by specifying a `type` in the `spec` of the Service: • *ClusterIP* (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster. Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/. **Service ClusterIP allocation** In Kubernetes, Services are an abstract way to expose an application running on a set of Pods. Services can have a cluster-scoped virtual IP address (using a Service of `type: ClusterIP`). Clients can connect using that virtual IP address, and Kubernetes then load-balances traffic to that Service across the different backing Pods. Source: https://kubernetes.io/docs/concepts/services-networking/cluster-ip-allocation.